PARLE CLAIR

CERTIFIED TRANSLATION OF DOCUMENTS

We, the undersigned, Parleclair, 1-3, Boulevard Charles de Gaulle, 92700 Colombes Cedex hereby certify that we are duly authorized to translate the French language, and have produced an accurate and exact translation in English of the French patent: PCT/FR0000130 – 20/01/2000, to the best of our translators' knowledge and skill.

Established in Colombes on October, 14th, 2006

1 bd Charles de Gaulle
92707 Colombes cedex
Tél : +33 1 41 45 05 75
Fax : +33 1 41 45 05 80
adv@parleclair.com
www.parleclair.com

SAS de 53 664 € - RC Nanterre 96 B 0358 - SIRET 775 669 906 00056 - APE 748F - N° TVA intracommunautaire FR 03 775 669 906

A COUNTERMEASURE METHOD IN AN ELECTRONIC COMPONENT USING A SECRET KEY CRYPTOGRAPHY ALGORITHM

The present invention concerns a countermeasure method in an electronic component using a secret key cryptography algorithm. They are used in applications where access to services or to data is strictly controlled. Such components have an architecture formed around a microprocessor and memories, including a program memory which contains the secret key.

These components are notably used in smart cards, for certain applications thereof. These are for example applications involving access to certain databanks, banking applications, remote payment applications, for example for television, petrol dispensing or passing through motorway tolls.

These components or cards therefore use a secret key cryptography algorithm, the best known of which is the DES algorithm (standing for Data Encryption Standard in British and American literature). Other

secret key algorithms exist, such as the RC5 algorithm or the COMP128 algorithm. This list is of course not exhaustive.

In general terms and briefly, the function of these algorithms is to calculate an enciphered message from a message applied at the input (to the card) by a host system (a server, banking dispenser etc) and the secret key contained in the card, and supplying this enciphered message in return to the host system, which for example enables the host system to authenticate the component or the card, to exchange data etc.

The characteristics of the secret key cryptography algorithms are known: calculations made, parameters used. The only unknown is the secret key contained in program memory. All the security of these cryptography algorithms relates to this secret key contained in the card and unknown to the world outside this card. This secret key cannot be deduced solely from knowledge of the message applied as an input and the enciphered message supplied in return.

However, it has become apparent that external attacks, based on current consumptions or a differential current consumption analysis when the microprocessor of a card is running the cryptography algorithm in order to calculate an enciphered message, enable ill-intentioned third parties to find the secret key contained in this card. These attacks are referred to as DPA attacks, the English acronym for Differential Power Analysis.

The principle of these DPA attacks is based on the fact that the current consumption of the microprocessor executing the instructions varies according to the data being manipulated.

Notably, when an instruction executed by the microprocessor requires manipulation of data bit by bit, there are two different current profiles depending on whether this bit is "1" or "0". Typically, if the microprocessor manipulates a "0", there is at this time of execution a first consumed current amplitude, and if the microprocessor manipulates a "1" there is a second consumed current amplitude, different from the first.

Thus the DPA attack exploits the difference in current consumption profile in the card during the execution of an instruction according to the value of the bit manipulated. In simplified terms, conducting a DPA attack consists of identifying one or more particular periods during which the algorithm is run comprising the execution of at least one instruction manipulating data bit by bit; reading a very large number N of current consumption curves during this period or periods, one curve per different message to which the algorithm is applied; predicting, for each curve, the value taken by a bit of the data for an assumption on a subkey, that is to say on at least part of the secret key, which makes it possible to make the prediction; and making a sort of the curves according to the corresponding Boolean selection function: a first packet of curves is obtained for which the prediction is "1" and a second packet of curves for

which the prediction is "0". By making a differential analysis of the mean current consumption between the two packets of curves obtained, an information signal DPA(t) is obtained. If the subkey assumption is not correct, each packet in reality comprises as many curves corresponding to the manipulation of a "1" as there are curves manipulating a "0". The two packets are therefore equivalent in terms of current consumption and the information signal is substantially zero. If the subkey assumption is correct, one packet actually comprises the curves corresponding to the manipulation of a "0" and the other packet actually comprises the curves corresponding to the manipulation of a "0": the information signal DPA(t) is not zero: it comprises consumption peaks corresponding to the manipulation by the microprocessor of the bit on which the sorting is based. These peaks have an amplitude corresponding to the difference in consumption by the microprocessor depending on whether it is manipulating a "1" or a "0". Thus, step by step, it is possible to discover all or part of the secret key contained in an electronic component.

There are many secret key algorithms for the execution of which the microprocessor must at certain times manipulate data bit by bit.

Notably, the algorithms generally comprise permutations which require such manipulations by the microprocessor. By analysing the current consumption during the execution of these manipulations bit by bit, it is possible to find the value of some bits at least

of the data item manipulated. Knowledge of this data item can supply information on intermediate results obtained during the execution of the enciphering algorithm, which in their turn can make it possible to find at least some of the bits of the secret key used.

The object of the present invention is to protect the data on which manipulations are made bit by bit, by applying a countermeasure to them, that is to say a scrambling, so that the analysis of the current consumption during the manipulation of this data reveals no information on this data: the information signal DPA(t) will always be zero whatever the subkey or key assumptions made in the DPA attacks.

As claimed, the invention concerns a countermeasure method in an electronic component using a cryptographic algorithm with a secret key K.

According to the invention, the countermeasure method consists, for an operation or a series of operations applied to an input data item and comprising at least one manipulation bit by bit, of first drawing a first random data item of the same size as the first data item, calculating a second random data item by effecting an exclusive OR between the first random data item and the input data item, and successively applying the operation or series of operations to the first random data item and to the second random data item.

In this way, the operation or series of operations manipulates only random data items so that it is no longer possible to implement a DPA attack.

In order to find the output data item corresponding to the application of the series of steps to the input data item, it suffices to calculate the exclusive OR between the first and second random results.

In a first method of applying this countermeasure method, the operation or series of operations relate to a data item calculated from the message to be enciphered.

In a second method of applying the countermeasure method according to the invention, this method is applied to operations relating directly to the secret key and supplying, for each round of the algorithm, the subkey to be used.

In this method of applying the countermeasure method according to the invention, provision is made for effecting a first series of steps according to the method indicated above so that a first random subkey and a second random subkey are obtained.

In this variant, instead of calculating the true subkey for the round in question, these random subkeys are used, so that the true subkey of each round no longer appears in clear: only random subkeys are manipulated.

Other characteristics and advantages of the invention are detailed in the following description given for indication and in no way limitatively, and with reference to the accompanying drawings, in which:

- Figures 1 and 2 are detailed flow diagrams of the first and second rounds of the DES algorithm;

- Figure 3 depicts schematically the countermeasure method according to the invention applied to an operation effecting a data manipulation bit by bit;

- Figure 4 depicts a first method of applying the countermeasure method according to the invention in the execution of the DES algorithm;

- Figure 5 depicts schematically a second method of applying the method according to the invention to the operations of the DES algorithm manipulating the secret key; and

- Figure 6 depicts a detailed flow diagram of the DES algorithm in an application of the countermeasure method corresponding to the diagram in Figure 5; and

- Figure 7 depicts a block diagram of a smart card in which it is possible to implement the countermeasure method according to the invention.

The DES secret key cryptographic algorithm (hereinafter reference will be made more simply to the DES or to the DES algorithm) includes 16 calculation rounds, denoted T1 to T16, as depicted in Figures 1 and 2.

The DES begins with an initial permutation IP on the input message M (Figure 1). The input message M is a word f of 64 bits. After permutation, a word e of 64 bits is obtained, which is divided into two in order to form the input parameters L0 and R0 of the first round (T1). L0 is a word d of 32 bits containing the 32 most significant bits of the word e. R0 is a word h of 32

bits containing the 32 least significant bits of the word e.

The secret key K, which is a word q of 64 bits, itself undergoes a permutation and a compression in order to supply a word r of 56 bits.

The first round comprises an operation EXP PERM on the parameter R0, consisting of an expansion and a permutation, in order to supply as an output a word 1 of 48 bits.

This word 1 is combined with a parameter K1, in an operation of the exclusive OR type denoted XOR, in order to supply a word b of 48 bits. The parameter K1, which is a word m of 48 bits, is obtained from the word r by a shift by one position (the operation denoted SHIFT in Figures 1 and 2) supplying a word p of 48 bits, to which an operation is applied comprising a permutation and a compression (the operation denoted COMP PERM).

The word b is applied to an operation denoted SBOX, at the output of which a word a of 32 bits is obtained. This particular operation consists of supplying an output data a taken from a table of constants TC_0 according to an input data item b.

The word a undergoes a permutation P PERM, giving as an output the word c of 32 bits.

This word c is combined with the input parameter L0 of the first round T1, in a logic operation of the exclusive OR type, denoted XOR, which supplies as an output the word g of 32 bits.

The word h (=R0) of the first round supplies the input parameter L1 of the following round (T2) and the word g of the first round supplies the input parameter R1 of the following round. The word p of the first round supplies the input r of the following round.

The other rounds T2 to T16 occur in a similar fashion, except with regard to the shift operation SHIFT, which is effected on one or two positions according to the rounds in question.

Each round Ti thus receives as an input the parameters Li-1, Ri-1 and r and supplies as an output the parameters Li and Ri and r for the following round Ti+1.

At the end of the DES algorithm (Figure 4), the enciphered message is calculated from the parameters L16 and R16 supplied by the last round T16.

This calculation of the enciphered message C comprises in practice the following operations:

- forming a word e' of 64 bits by reversing the positions of the words L16 and R16 and then concatenating them;

- applying the permutation IP^{-1} which is the reverse of that of the start of DES, in order to obtain the word f' of 64 bits forming the enciphered message C.

It can be seen that this algorithm comprises many operations manipulating the data bit by bit, like the permutation operation.

According to the countermeasure method according to the invention, a software countermeasure is applied

when the microprocessor which calculates the enciphered message effects a manipulation bit by bit. In this way, the statistical processing and the Boolean selection function of the DPA attack applied to the current consumption curves no longer supplies any information: the signal DPA(t) remains zero whatever the subkey assumptions made.

The software countermeasure according to the invention then consists of making each of the bits manipulated by the microprocessor unpredictable.

The principle of this countermeasure is depicted in Figure 3.

Let an input data item be D.

Let there be an operation OPN to be calculated on this input data item D, the result of which is denoted OPN(D). This operation OPN requires a bit by bit manipulation of the input data item D by the microprocessor; it is a case for example of a permutation.

According to the invention, instead of applying the operation OPN to the input data item D in order to calculate the result OPN(D) of the operation, the following different steps are performed:

- drawing a random value for a first random data item U, of the same size as the input data item D (for example 32 bits);

- calculating a second random data item V by effecting an exclusive OR between the input data item and the first random data item: V = D XOR U;

- calculating the operation OPN on the first random data item U, giving a first random result OPN(U);

- calculating the operation OPN on the second random data item V, giving a second random result OPN(V);

- calculating the result OPN(D) by effecting an exclusive OR between the first and second random results: OPN(D) = OPN(U) XOR OPN(V).

This method can equally well be applied to a single operation or to a series of operations.

A first method of applying the countermeasure method according to the invention concerns operations on data calculated from the message (M) to which the algorithm is applied. The input data item D is in this case a data item calculated from the message M.

In a practical example of this first method of application of the algorithm DES depicted in Figure 4, this method is applied on the one hand to the operation EXP PERM and on the other hand to the operation P PERM, which both comprise a permutation requiring a bit by bit manipulation of the input data item.

In the figure the application of this countermeasure to these operations is denoted CM(EXP PERM) and CM(P PERM).

The software countermeasure according to the invention then consists of performing, in place of each operation P PERM and EXP PERM, the operations CM(EXP PERM) and CM(P PERM) according to the calculation sequence described in Figure 3, using a random variable

U. As each round of the algorithm comprises an operation EXP PERM and an operation P PERM, this countermeasure can be applied in each of the rounds of the DES.

Experience shows that it is the first three rounds and the last three rounds which allow DPA attacks. Afterwards, it becomes very difficult or even impossible to predict the bits.

Thus an implementation of a countermeasure method according to the invention which is less expensive in calculation time consists of applying only these first three and last three rounds of the DES.

Different variant applications of the countermeasure method according to the invention concern the drawing of a random value for the first random data item U. Depending on whether or not a great deal of calculation time is available, it is possible to draw a new random value each time, for each of the operations or series of operations for which the countermeasure method according to the invention is implemented.

Thus, in Figure 4, for the operation CM(EXP PERM), a value u1 for the random data item U is drawn and, for the operation CM(P PERM), another value u2 is drawn for the random value U.

Or else it is possible to draw a new random value for each round of the algorithm, or a single random value at the start of the algorithm.

The implementation of the countermeasure method according to the invention depends principally on the

applications concerned, depending on whether or not it is possible to devote a great deal of additional time to the countermeasure.

A second mode of applying the countermeasure method according to the invention is depicted in Figure 5. It concerns more particularly the calculation operations applied to the secret key K in order to supply each of the subkeys Ki used in the rounds of the algorithm. In the example of the DES, these operations are the following KEY PERM, executed at the start of DES and SHIFT and COMP PERM executed at each round. During these operations, at certain times, the microprocessor separately manipulates a bit of the secret key, therefore leaving the possibility of a DPA attack on this bit.

The countermeasure method according to the invention is then applied by protecting the data item, the secret key in this case, before performing these operations, so that it is no longer possible to obtain information by DPA attack.

Thus, and as schematically shown in Figure 5, a random value of a first random data item Y is drawn, with the same size as the secret key K. A second random data item Z with the same size is calculated, making an exclusive OR between the secret key K and the first random data item Y: Z = K XOR Y.

In the example, the sequence of operations comprises the following operations KEY PERM, SHIFT, COMP PERM. Then this sequence of operations is applied to each of the two random data items Y and Z,

successively. Thus, from these two data items Y and Z applied successively as an input, the data items Y', $P_{iY'}$, $K_{iY'}$, or respectively Z', $P_{iZ'}$, $K_{iZ'}$ are obtained, at the output of the operations KEY PERM, SHIFT, COMP PERM.

A practical example of an application to the DES is shown in Figure 6.

In the DES, the operation KEY PERM is executed only once, at the start, whilst the sequence of operations SHIFT and COMP PERM is executed in each round.

In addition, the output of the operation SHIFT of a round Ti is applied as an input of the operation SHIFT of the following round Ti+1 (see Figures 1 and 2).

In order to apply the countermeasure method according to the second mode of application to this DES algorithm, the first operation KEY PERM is then applied to the random data Y and Z, which gives two intermediate random data, denoted Y' and Z'. These two intermediate random data are successively applied to the operations SHIFT of the first round T1, supplying two intermediate random data denoted $P_{1Y'}$ and $P_{1Z'}$. These two random data are on the one hand stored in working memory for the operation SHIFT of the following round (the second round), and on the other hand applied successively to the operation EXP PERM of the first round, in order to supply a first intermediate result $K_{1Y'}$ and $K_{1Z'}$.

This procedure is followed in each round. Thus, at each round Ti, a first random result is obtained: $K_{iY'}$ = EXP PERM (SHIFT (Y')) and a second random result: $K_{iZ'}$ = EXP PERM (SHIFT (Z'));

and the intermediate random data SHIFT (Y')=$P_{iY'}$ and SHIFT (Z')=$P_{iZ'}$ are stored in working memory for the following round Ti+1.

For each round Ti, it would then be possible to recalculate the corresponding subkey Ki corresponding to the sequence of operations KEY PERM, SHIFT and COMP PERM of this round applied to the secret key K, making an exclusive OR between the two random results $K_{iY'}$ and $K_{iZ'}$: Ki = $K_{iY'}$ XOR $K_{iZ'}$.

However, preferably and as depicted in Figure 6, the subkey K_i of the round Ti is not recalculated. The first random result $K_{iY'}$ is applied in place of the subkey Ki in an exclusive OR operation XOR with the data item 1 supplied by the permutation expansion operation EXP PERM. An intermediate result b' is obtained.

By then effecting an exclusive OR XOR of this intermediate result b' with the second random result $K_{iZ'}$, the output data item b = XOR (1, Ki) is found. The following operations are then performed in each round Ti, in order to calculate the parameter b from 1:

b' = 1 XOR $K_{iY'}$ and

b = b' XOR $K_{iZ'}$, as shown for the first and second rounds in Figure 6.

In this way, the secret subkey itself is no longer used in calculating the enciphered message, but

"random subkeys": the key is then protected before and during the execution of the cryptographic algorithm, since $K_{iY'}$ and $K_{iZ'}$ being random and not known to the external world of the component (or of the card), they are liable to change at each new execution of the cryptography algorithm. It should be noted that, in the application of the countermeasure method according to the invention to the calculation and use of the subkeys, a random value is drawn only once, at the start of execution of the algorithm, before the operations on the secret key.

This second mode of applying the countermeasure method according to the invention to the secret key can advantageously be combined with the first mode of applying the countermeasure method to the calculation of the enciphered message proper, this combination making the countermeasure particularly effective.

The present invention applies to the DES secret key cryptography algorithm, for which examples of implementation have been described. It applies more generally to any secret key cryptography algorithm where the execution by the microprocessor of certain operations requires a bit by bit manipulation of data.

An electronic component 1 using a countermeasure method according to the invention in a DES secret key cryptography algorithm comprises typically, as shown in Figure 7, a microprocessor oP, a program memory 2 and a working memory 3. Means 4 of generating a random value are provided which, if reference is made to the flow diagrams in Figures 3 and 5, will supply the random

values U and/or Y of the required size (32 bits for U, 64 bits for Y) at each execution of the cryptography algorithm. Such a component can particularly be used in a smart card 5, in order to improve its resistance to tampering.

CLAIMS

1. A countermeasure method in an electronic component using a cryptographic algorithm with a secret key K on an input message (M), characterised in that the execution of an operation (OPN) or of a sequence of operations comprising a bit by bit manipulation of an input data item (D), in order to supply an output data item (OPN(D)), comprises the following steps:

- drawing a random value, of a first random data item (U), with the same size as the input data item (D);

- calculating a second random data item (V), effecting an exclusive OR between the input data item and the first random data item (U);

- executing the operation (OPN) or the sequence of operations following on from the first random data item (U) and the second random data item (V), supplying respectively a first random result (OPN(U)) and a second random result (OPN(V)).

2. A countermeasure method according to Claim 1, also comprising the following step:

- calculating the output data item (OPN(D)) effecting an exclusive OR between the first and second random results.

3. A countermeasure method according to Claim 1 or 2, characterised in that it is applied to operations (EXP PERM, P PERM) relating to data calculated from the input message (M).

4. A countermeasure method according to any one of the preceding claims, characterised in that a new

random value (U) is drawn at each new execution of the said operation or sequence of operations.

5. A countermeasure method according to Claim 1, applied to an operation or a sequence of operations (KEY PERM, SHIFT, COMP PERM) performed on the said secret key (K).

6. A countermeasure method according to Claim 5, the cryptography algorithm comprising several calculation rounds, and comprising a sequence of operations on the secret key K in order to supply, at each round, (T_i), a corresponding subkey (K_i), a method characterised in that it is applied to the said sequence of operations in order to supply, at each round, a first random result ($K_{iY'}$) and a second random result ($K_{iZ'}$).

7. A countermeasure method according to Claim 6, each round (Ti) an exclusive OR operation between the subkey (K_i) and an input data item (1) in order to supply an output data item (b), characterised in that this operation is replaced by the following operations:

- calculating the exclusive OR between the said input data item (1) and the first random result ($K_{iY'}$) in order to supply an intermediate result (b');

- calculating the exclusive OR between the said intermediate result (b') and the second random result ($K_{iZ'}$) in order to supply the said output data item (b).

8. A countermeasure method according to any one of Claims 1, 2, 3, 5, 6 and 7, characterised in that a new random value (U or Z) is drawn at each new execution of the cryptography algorithm.

9. A countermeasure method according to either one of Claims 3 and 4, characterised in that it is combined with a countermeasure method according to any one of Claims 5 to 8.

10. A countermeasure method according to any one of the preceding claims, characterised in that it is applied to the DES algorithm.

11. An electronic security component implementing the countermeasure method according to any one of the preceding claims, characterised in that it comprises means (4) of generating a random value.

12. A smart card comprising an electronic security component according to Claim 9.